DNS: Hệ thống Tên miền

Con người : có thể nhận dạng bằng nhiều cách:

- Số Chứng Minh Thư
- Tên, Biệt danh
- Số hộ chiếu

Máy tính và Router trên Internet

- Địa chỉ IP (32 bit) - sử dụng để đánh địa chỉ cho các datagram
- "Tên", ví dụ như gaia.cs.umass.edu

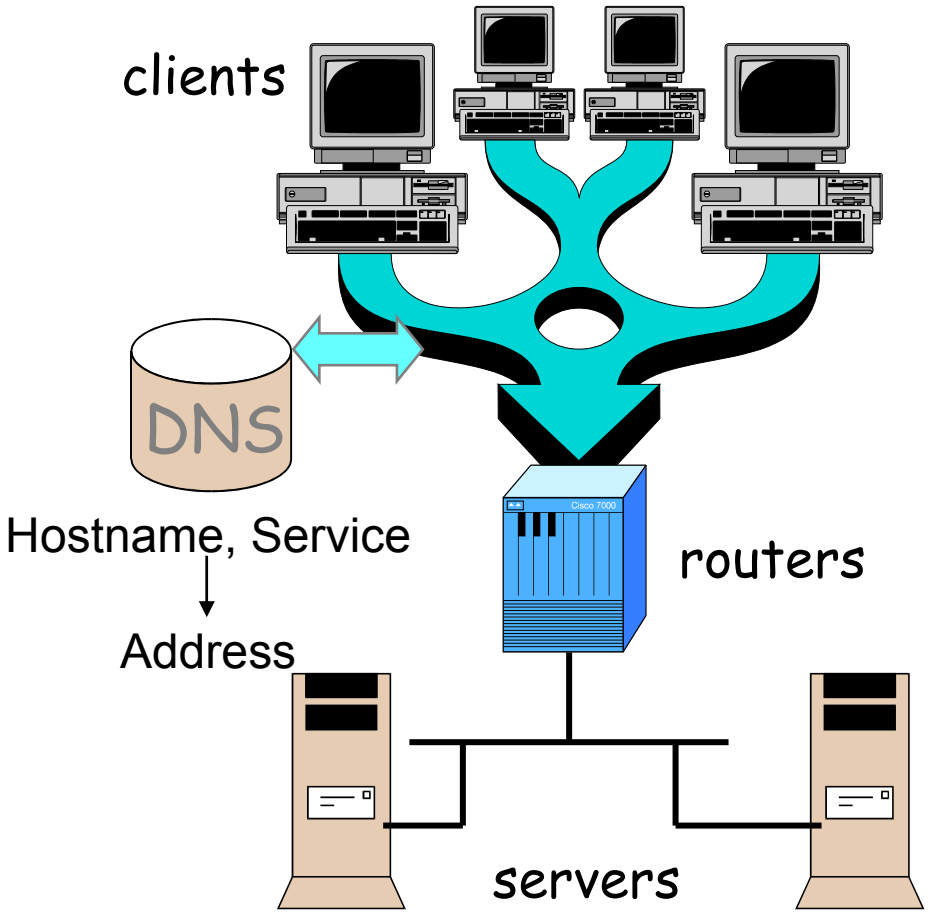
Q: Ánh xạ giữa Địa chỉ IP và Tên?

Domain Name System:

- Là **Hệ cơ sở dữ liệu phân tán** cài đặt bởi nhiều **name servers** phân cấp
- **Giao thức tầng ứng dụng** : host, router yêu cầu name servers để giải mã tên (ánh xạ địa chỉ <->tên)
  - Chú ý : Chức năng cơ bản của Internet hoạt động như giao thức tầng Ứng dụng
  - "Phức tạp" đặt ở "rìa"

DNS: Hệ thống tên miền

- Chức năng
  - Ánh xạ (Tên miền, dịch vụ) đến một giá trị nào đó
    - (www.cs.yale.edu, Addr) -> 128.36.229.30
    - (cs.yale.edu, Email) -> netra.cs.yale.edu
    - (netra.cs.yale.edu, Addr) -> 128.36.229.21
- Tại sao không sử dụng địa chỉ IP trực tiếp ?



DNS: Name Server (Máy chủ Tên)

Tại sao không nên tạo ra một DNS Server tập trung ?

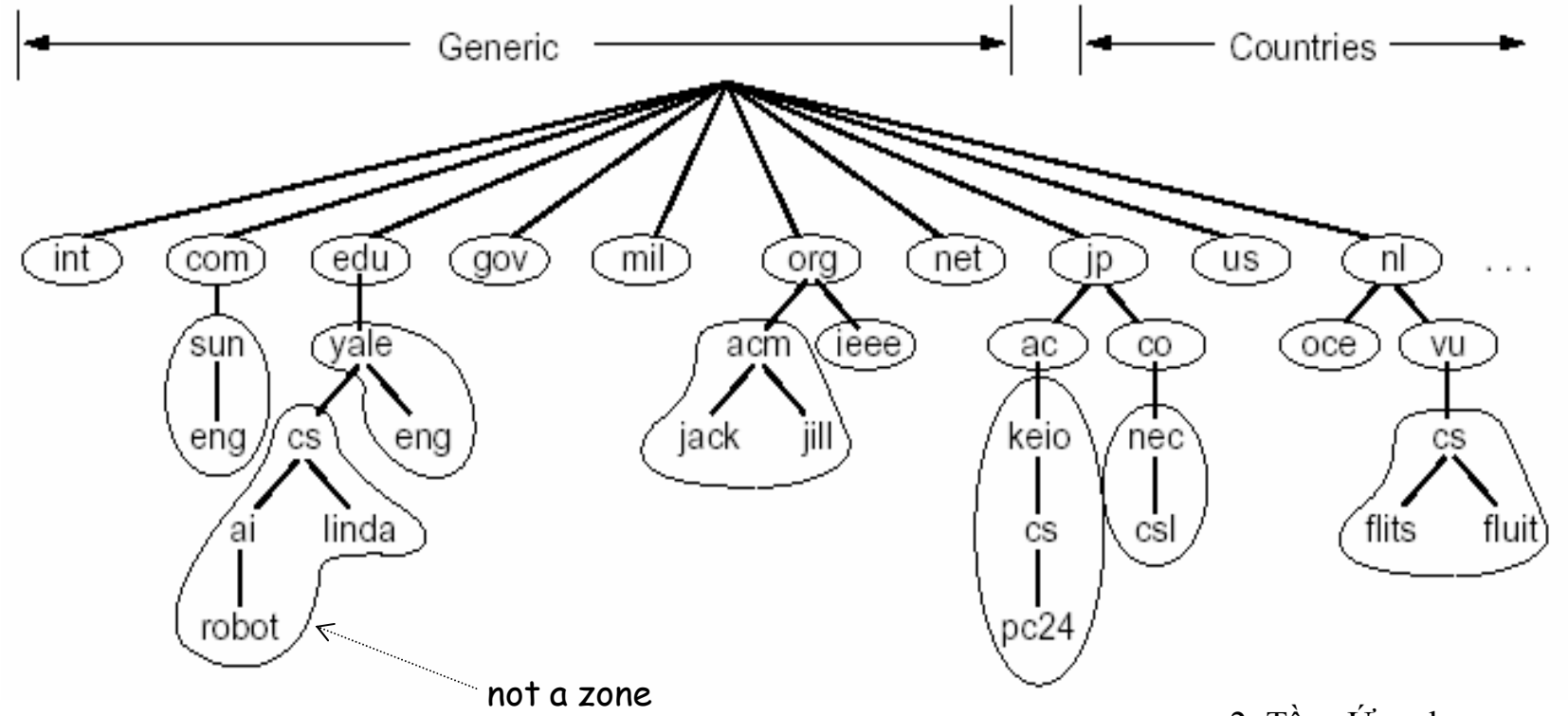
- Điểm hỏng duy nhất - nếu name-server "chết" thì cả mạng Internet sẽ "chết" theo.
- Khối lượng giao dịch tại điểm tập trung lớn.
- Cơ sở dữ liệu tập trung ở "xa" với nhiều nơi
- Bảo trì dễ hơn.

Không* MỞ RỘNG *được !

- **Local name servers:**
  - Mỗi ISP, công ty có *local name server (ngầm định)*
  - Câu hỏi truy vấn của host về DNS sẽ được chuyển tới local name server
- **Chức năng của name server:**
  - Đối với host: lưu địa chỉ IP và tên miền tương ứng của host
  - Có thể tìm tên miền ứng với địa chỉ IP và ngược lại

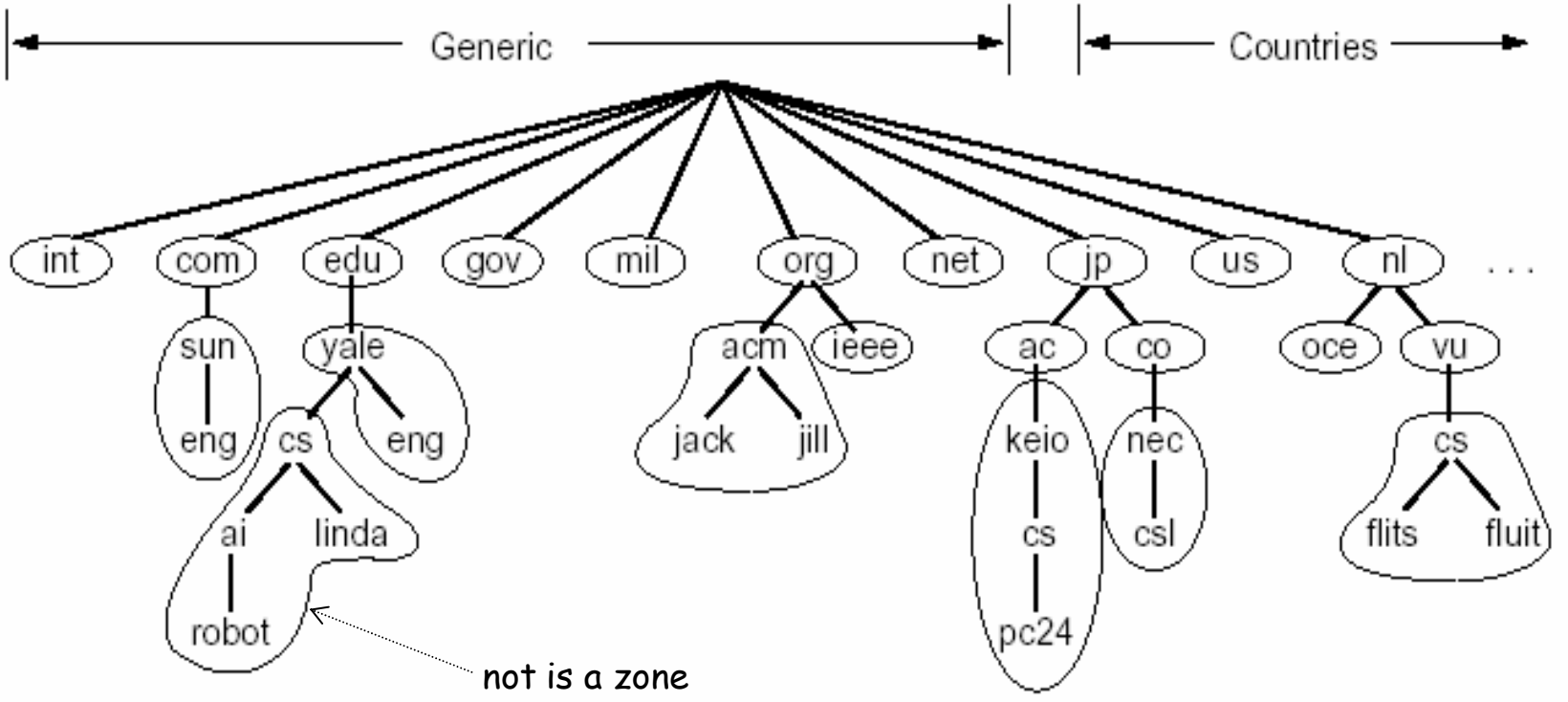
DNS: Đặt tên như thế nào ?

- Cơ chế Đặt tên
  - Không gian tên (phân cấp) được chia thành các vùng (zone)
  - Mỗi vùng có thể được coi là nhánh của cây tổng quát



Quản lý Phân tán Không gian Tên

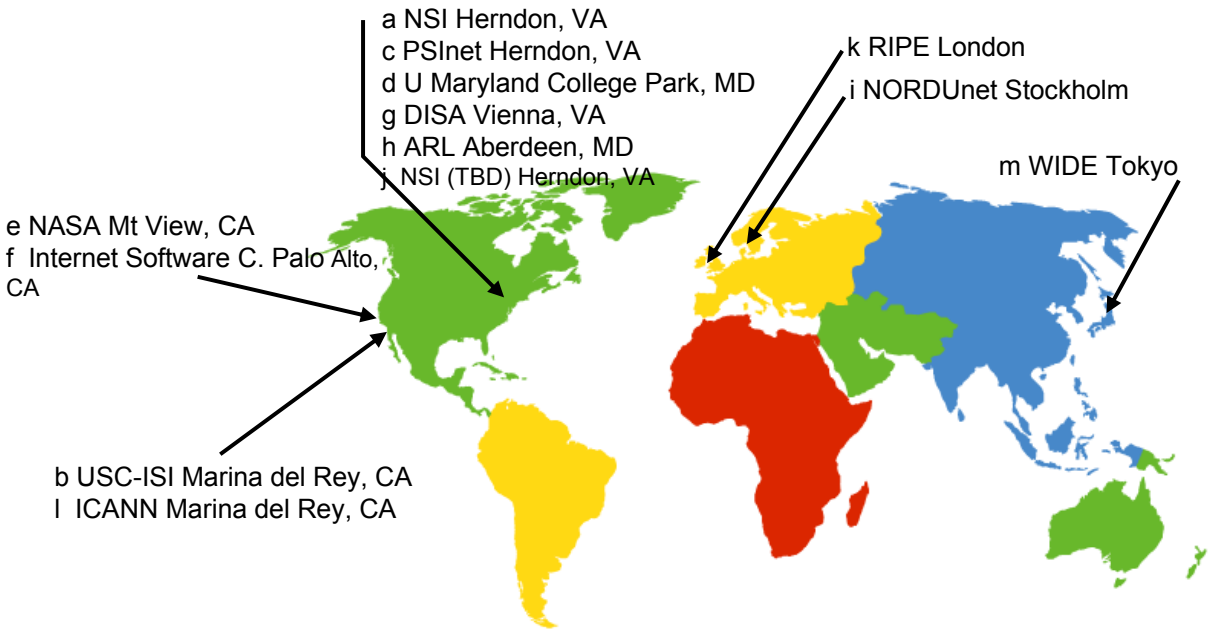
- Là cơ sở dữ liệu phân tán được nhiều authoritative name server quản lý
  - Mỗi Vùng có một **Authoritative Name Server** riêng
  - authoritative name server of a zone có thể **trao quyền quản lý** một bộ phận trong vùng của mình (tức là một nhánh con) cho name server khác



DNS: Root Name Server

- Local name server sẽ hỏi Root name server khi không xác định được ánh xạ.
- Root name server:
  - Đề nghị Local name server hỏi name server mức đỉnh

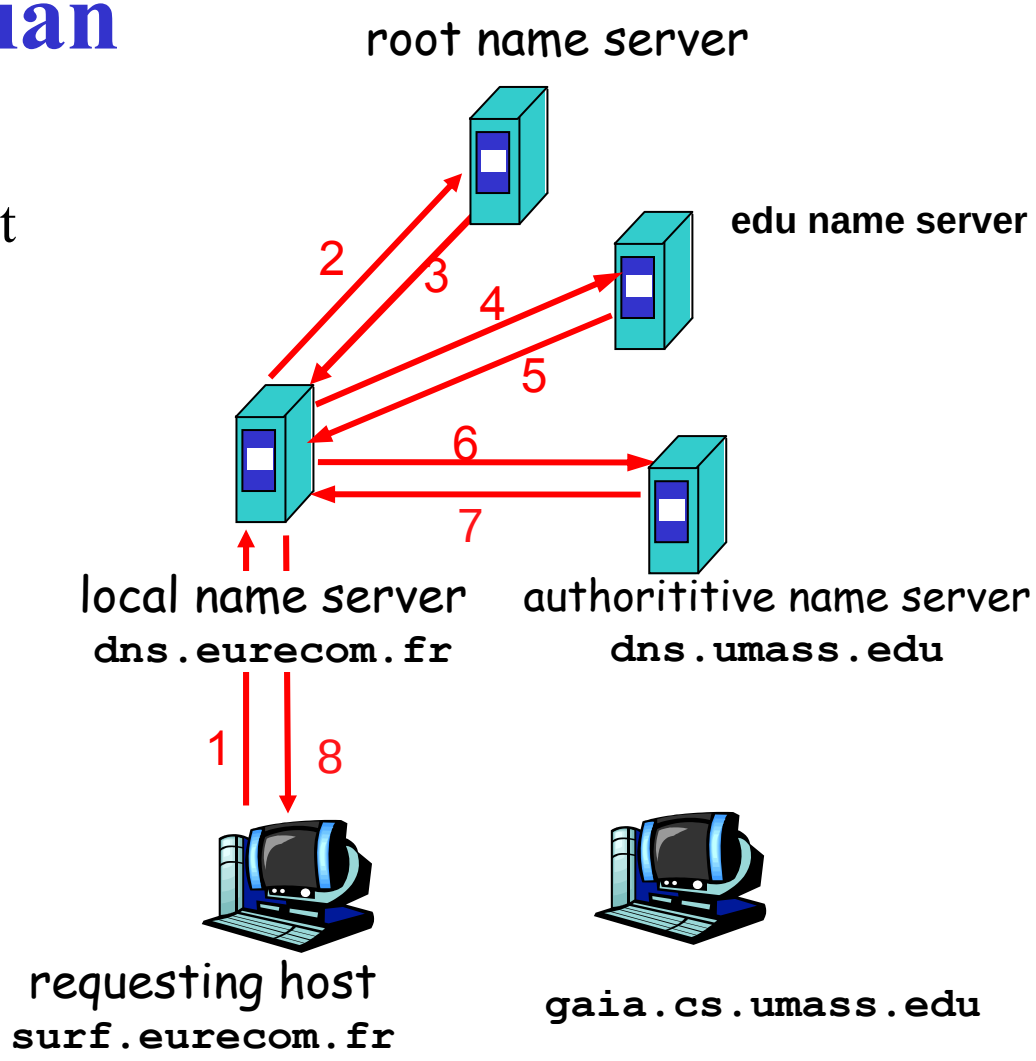
13 root name server trên Thế giới



DNS: Ví dụ đơn giản

host **surf.eurecom.fr** muốn biết địa chỉ IP của **gaia.cs.umass.edu**

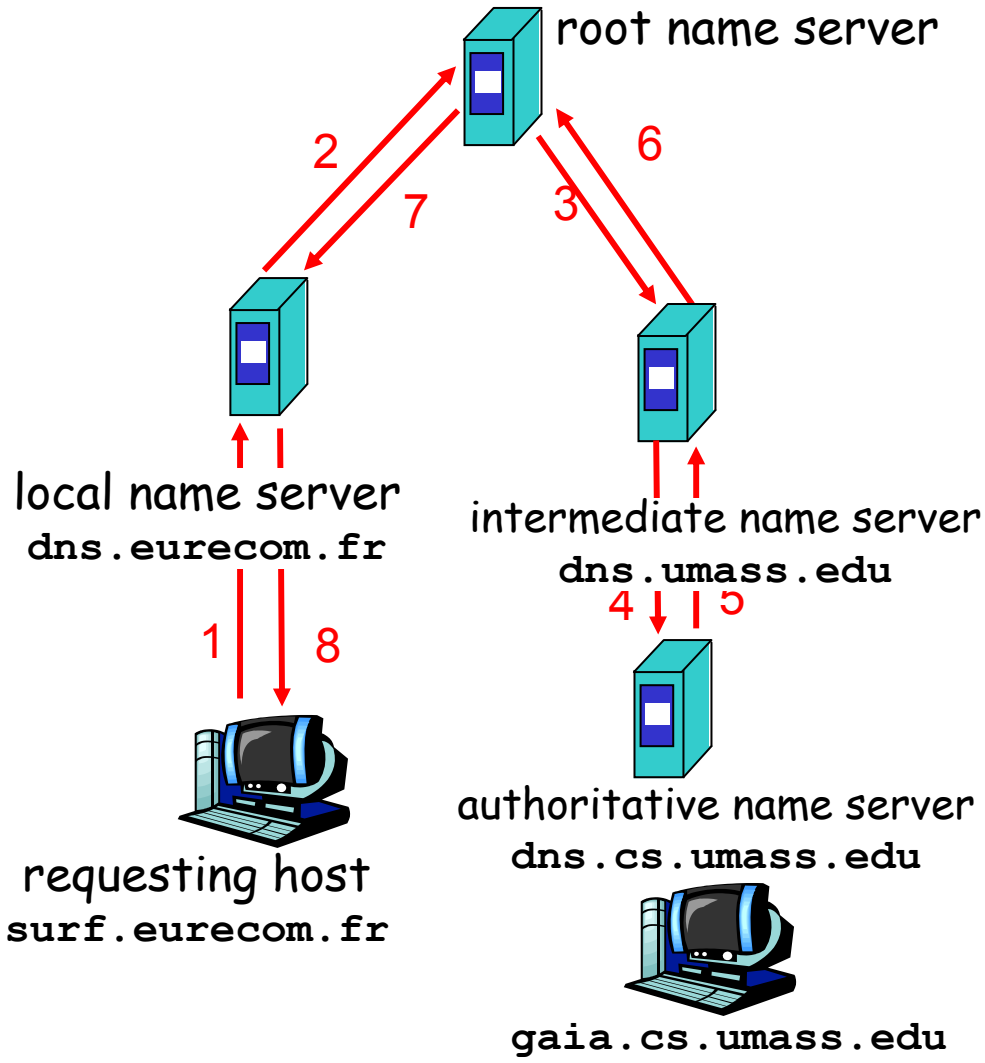
1. Hỏi local DNS server (**dns.eurecom.fr**)
2. **dns.eurecom.fr** hỏi root name server nếu cần thiết
3. root name server hỏi authoritative name server, **dns.umass.edu** nếu cần thiết.



Name Server Trung gian

Root name server:

- Có thể không biết authoritative name server
- Chỉ biết Name Server ***trung gian,*** qua đó mới tìm được authoritative name server



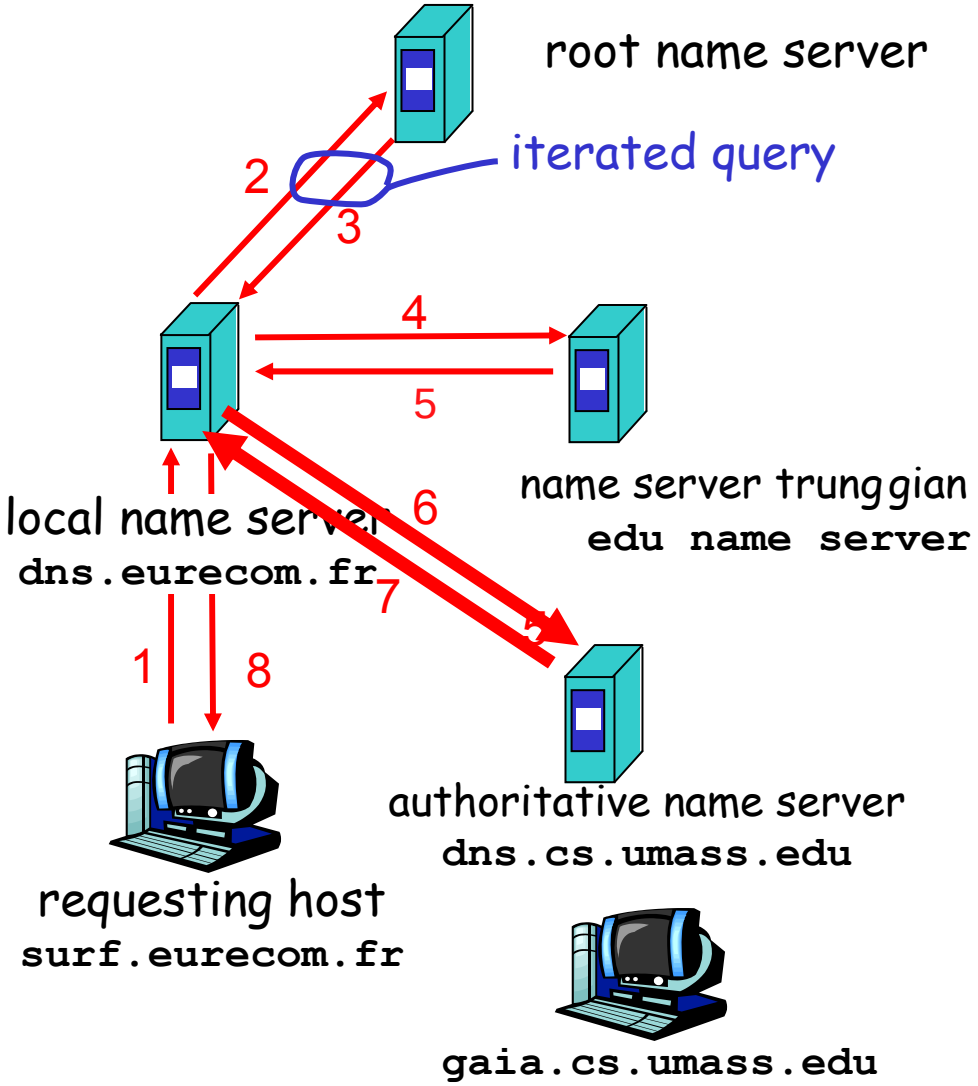
DNS: các loại truy vấn

Truy vấn đệ quy :

- Name server là nơi giải mã địa chỉ/tên. Nếu không tự mình giải mã được sẽ gửi yêu cầu đến name server khác.
- Root name server liệu có bị quá tải ?

Truy vấn lặp:

- Nếu không phân giải được địa chỉ IP, gửi thông điệp "*Tôi không biết, hãy hỏi bạn tôi là A*". A là địa chỉ IP của name server kế tiếp.



DNS: Lưu tạm và cập nhật bản ghi

- Khi "học" được thêm một ánh xạ, name server sẽ "ghi nhớ" ánh xạ này
  - Sau một khoảng thời gian, nếu thành phần nào trong cache không được sử dụng thì sẽ bị xóa bỏ.

- Cơ chế Cập nhật và Thông báo do IETF thiết kế:
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

Bản ghi DNS

DNS: cơ sở dữ liệu phân tán lưu các Bản ghi Tài nguyên (RR)

Định dạng RR : (name, value, type, TTL)

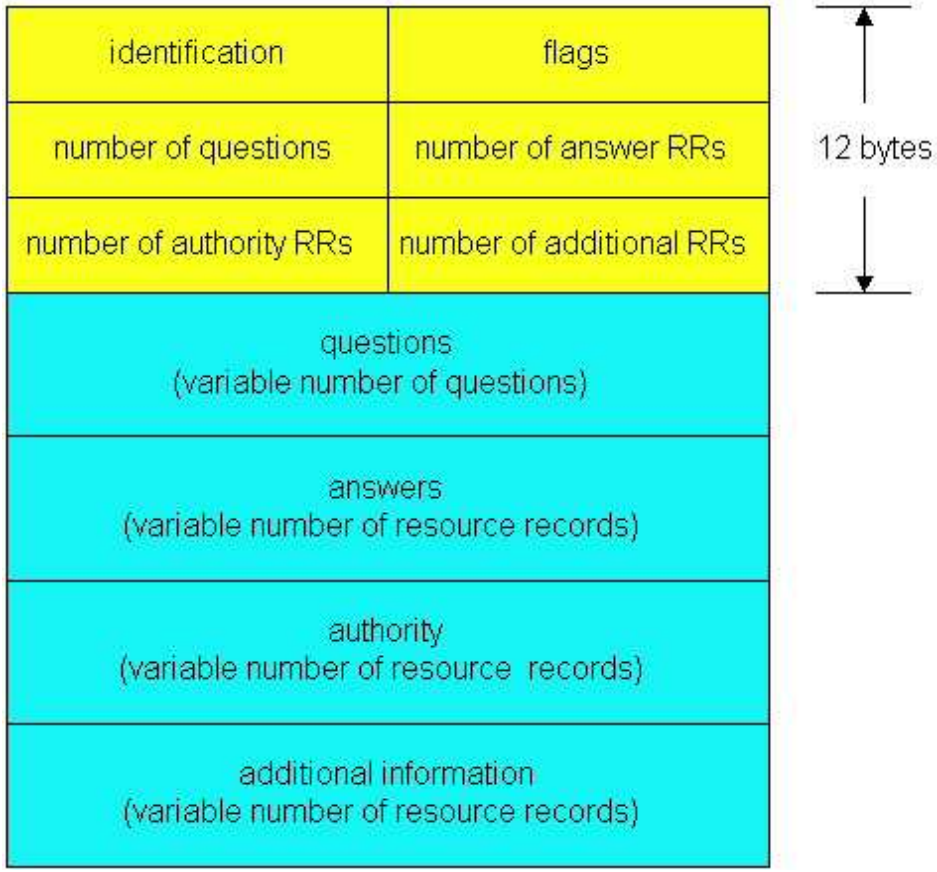
- **Type=A**
  - **name**: hostname
  - **value**: IP address
- **Type=NS**
  - **name**: domain (ví dụ foo.com)
  - **value**: địa chỉ IP của authoritative name server ứng với miền đó
- **Type=CNAME**
  - **name**: bí danh cho một tên thực nào đó: ví dụ *www.ibm.com* là bí danh của *servereast.backup2.ibm.com*
  - **value**: tên thực
- **Type=MX**
  - **value**: tên của mailserver

Giao thức và Thông điệp của DNS

Giao thức DNS : Thông điệp ***truy vấn*** và ***trả lời*** đều có chung một định dạng

Tiêu đề Thông điệp

- **identification:** 16 bit. Thông điệp trả lời và truy vấn có cùng định danh.
- **Cờ:**
  - Cờ query hay reply
  - Cờ mong muốn đệ quy
  - Cờ chấp nhận đệ quy
  - Cờ authoritative



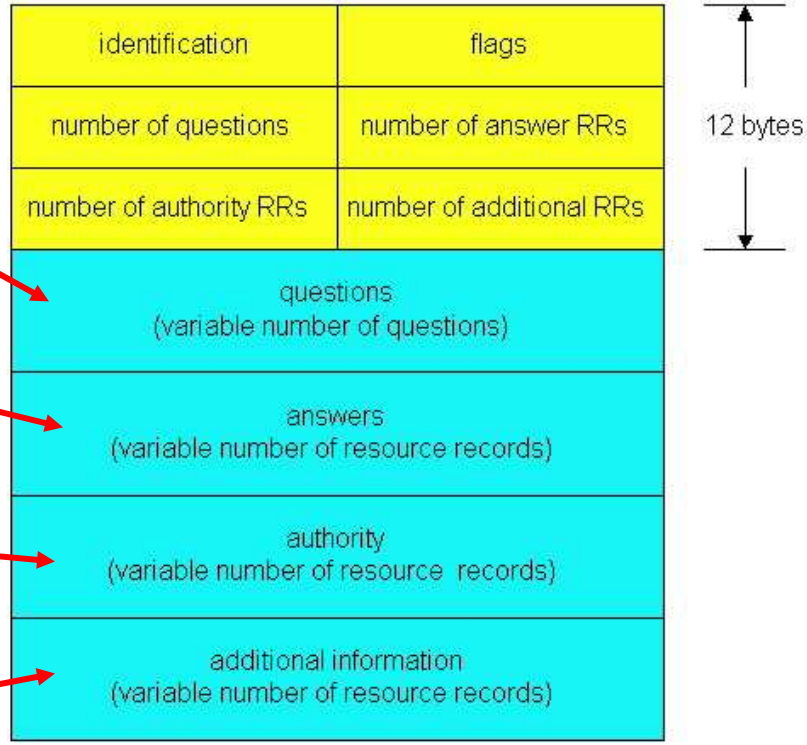
DNS – Giao thức và Thông điệp

Trường tên và Kiểu truy vấn

Thông điệp trả lời

Các bản ghi của authoritative server

Những thông tin ngoại lệ có thể hữu dụng



Quan sát DNS

- Sử dụng lệnh **dig** (Hoặc **nslookup**):
  - Yêu cầu Hỏi bằng câu hỏi tương tác:
    %dig +trace www.cnn.com

- Bắt thông điệp bằng Ethereal
  - DNS server lắng nghe ở port 53

DNS – Ưu điểm ?

- Phân cấp: nâng cao năng lực quản lý và tăng cường khả năng mở rộng
- Nhiều server: tăng khả năng phòng chống lỗi
  - Xem http://www.internetnews.com/dev-news/article.php/1486981 để thấy Tấn công từ chối Dịch vụ (DDoS) vào hệ thống root server vào tháng 10/2002 (9 trong 13 root server bị đình trệ, nhưng mạng chỉ bị chậm đi không đáng kể)
  - Xem http://www.cymru.com/DNS/index.html để thấy hiệu suất được giám sát như thế nào
- Caching làm giảm tải và giảm thời gian phản hồi

DNS – Nhược điểm

- Hệ thống Tên miền không phải là phương thức tốt nhất để đặt tên các tài nguyên khác, chẳng hạn file
- Số lượng giới hạn Kiểu tài nguyên hạn chế khả năng đưa thêm các dịch vụ mới
- Mặc dù về mặt lý thuyết có thể cập nhật bản ghi tài nguyên, nhưng trên thực tế hiếm khi làm được.
- Mô hình truy vấn đơn giản => khó cài đặt những dạng truy vấn phức tạp
- Kết nối sớm (Tách biệt truy vấn DNS với ứng dụng đưa ra truy vấn) không hiệu quả trong môi trường di động và thay đổi thường xuyên
  - Ví dụ : Cân bằng tải, Tìm máy in gần nhất

Giải pháp Phân giải Tên kiểu Linda

- Nhiều đề xuất dựa trên "Không gian làm việc Phân tán" (Linda) do David Gelernter đưa ra
  - Intentional Naming System (INS),
  - Internet Indirect Infrastructure (I3)

- Nút viết các các tuples (một dạng vector không kiểu) vào các "không gian dùng chung"
- Nút đọc các tuple phù hợp từ không gian dùng chung

Lập trình Socket

Mục đích : Nghiên cứu cách xây dựng ứng dụng client/server giao tiếp qua socket

Socket API

- BSD4.1 UNIX, 1981
- Ứng dụng Tạo, Sử dụng và Đóng socket một cách tường minh.
- Sử dụng theo mô hình Client/Server
- Hai kiểu dịch vụ ứng dụng sử dụng socket API:
  - Truyền không tin cậy
  - Tin cậy, hướng nối, đúng thứ tự.

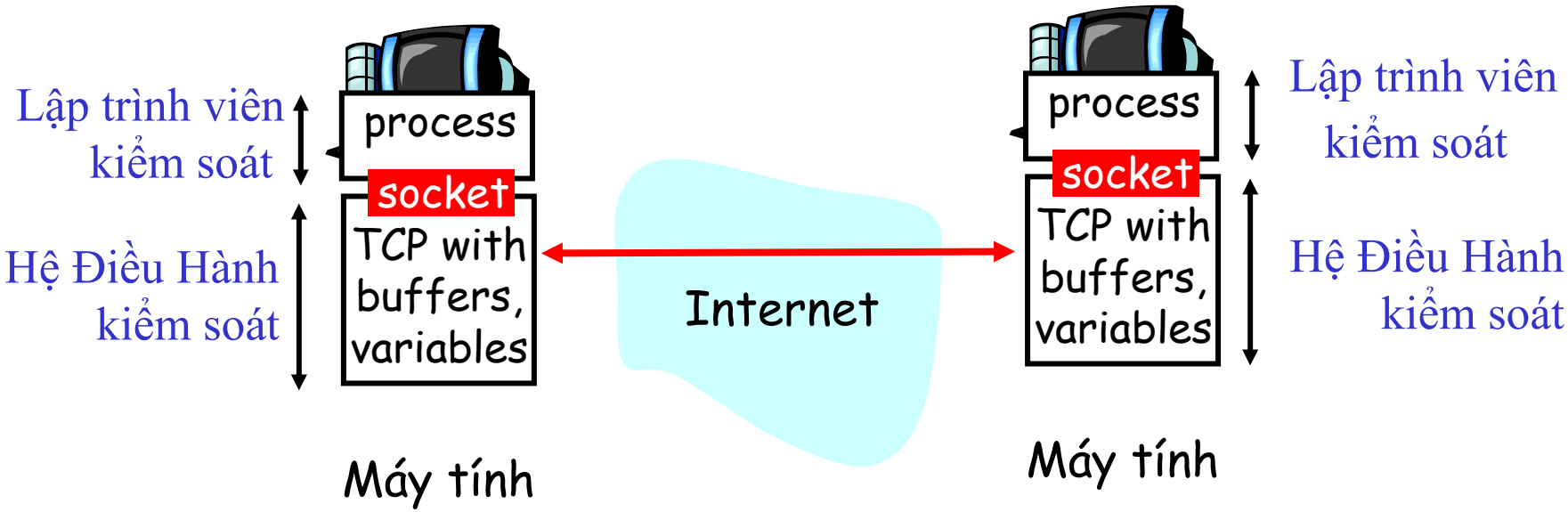
socket

Là giao diện *nằm trên máy tính,* do *ứng dụng tạo ra và quản lý,* nhưng *HĐH kiểm soát* (là "cửa") thông qua đó tiến trình vừa gửi và nhận thông điệp từ các tiến trình ứng dụng khác (ở trên máy tính khác)

Lập trình Socket TCP

Socket: Là "cửa" giữa tiến trình ứng dụng và giao thức giao vận đầu cuối (UDP/TCP)

TCP: Dịch vụ truyền byte tin cậy từ tiến trình này sang tiến trình khác.



Lập trình TCP Socket

Client phải liên lạc với server

- Tiến trình trên server phải chạy trước.
- Server phải tạo sẵn socket (door) để tiếp nhận yêu cầu từ client.

Client trao đổi với server bằng cách:

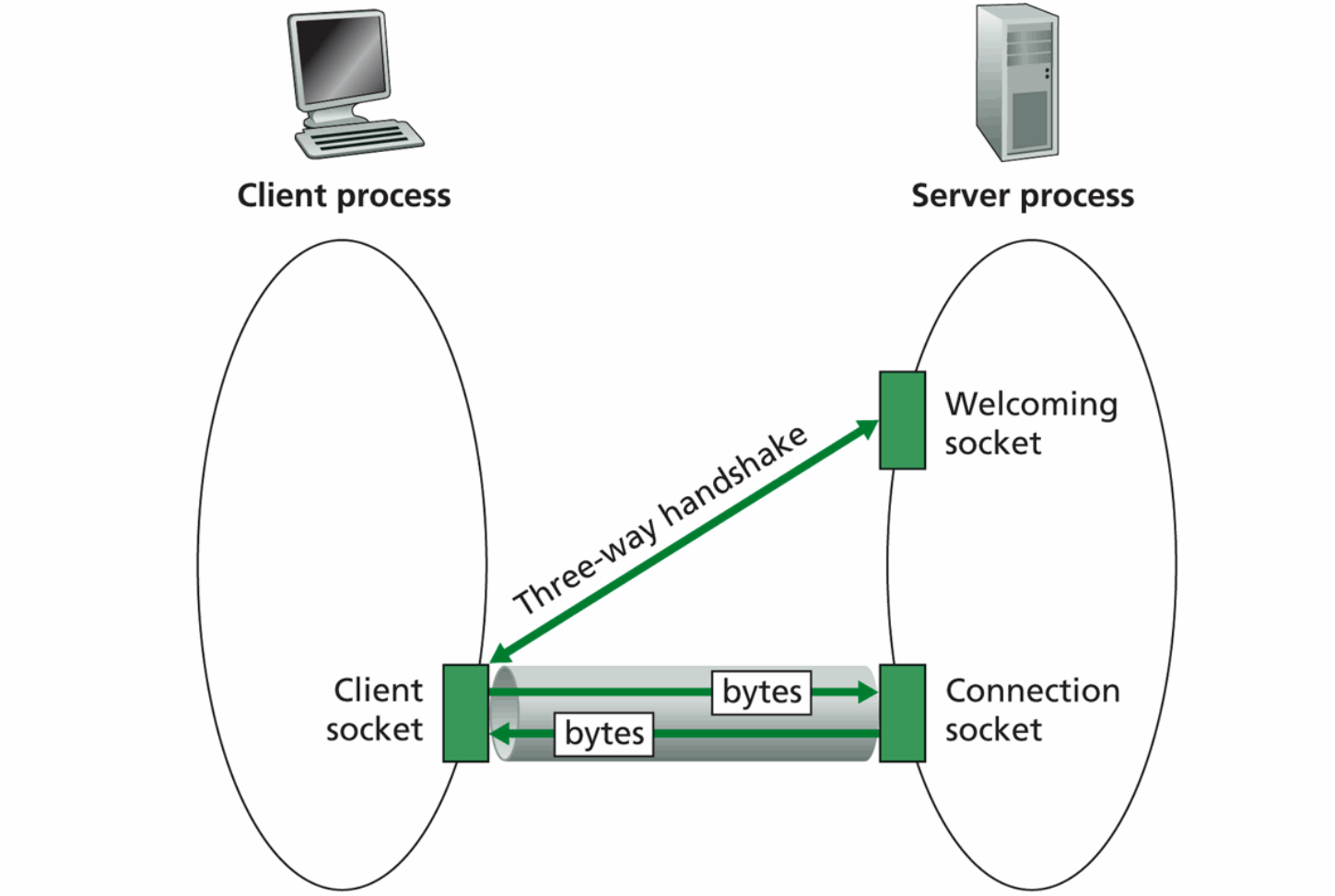
- Tạo socket TCP ở phía client
- Xác định địa chỉ IP, số hiệu cổng của tiến trình server.

- Khi client tạo socket: client TCP thiết lập kết nối tới server TCP.
- Khi nhận được yêu cầu từ client, server TCP tạo socket mới cho tiến trình trên server trao đổi dữ liệu với client
  - Cho phép server có thể đáp ứng yêu cầu của nhiều client.

Quan điểm Lập trình Ứng dụng

TCP cung cấp dịch vụ truyền dữ liệu tin cậy theo byte giữa Client và Server

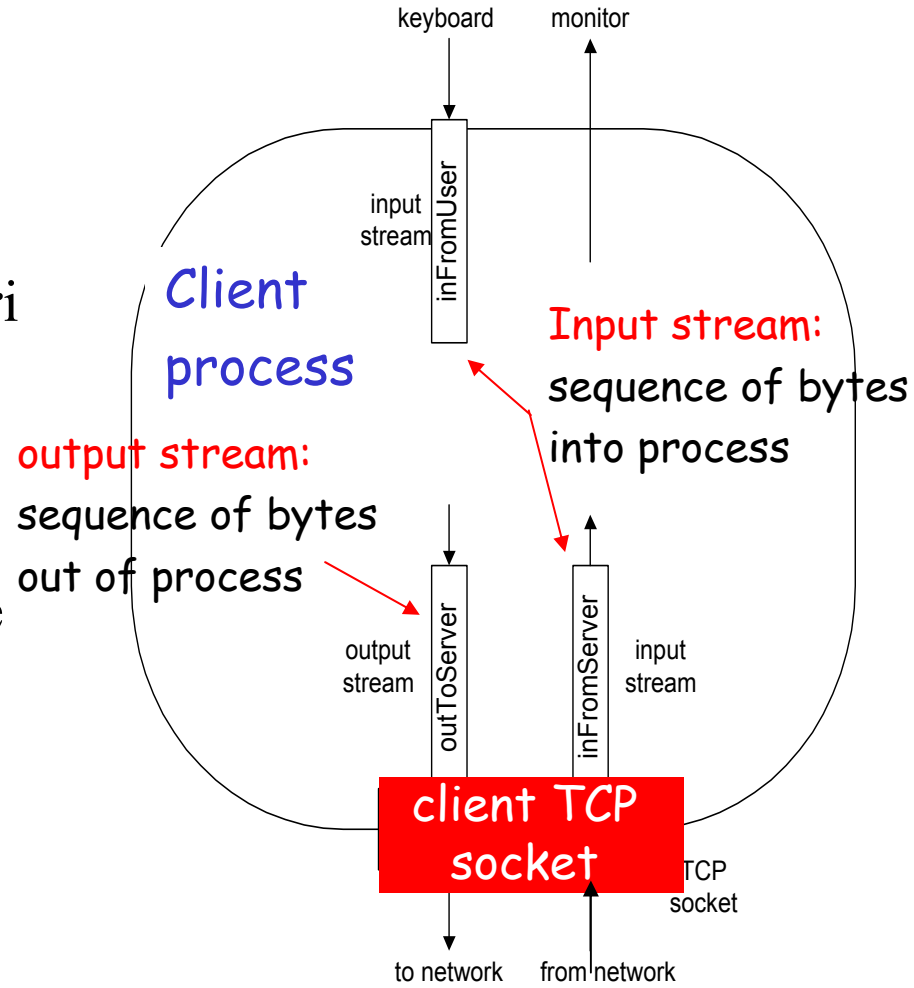
Hướng nối TCP



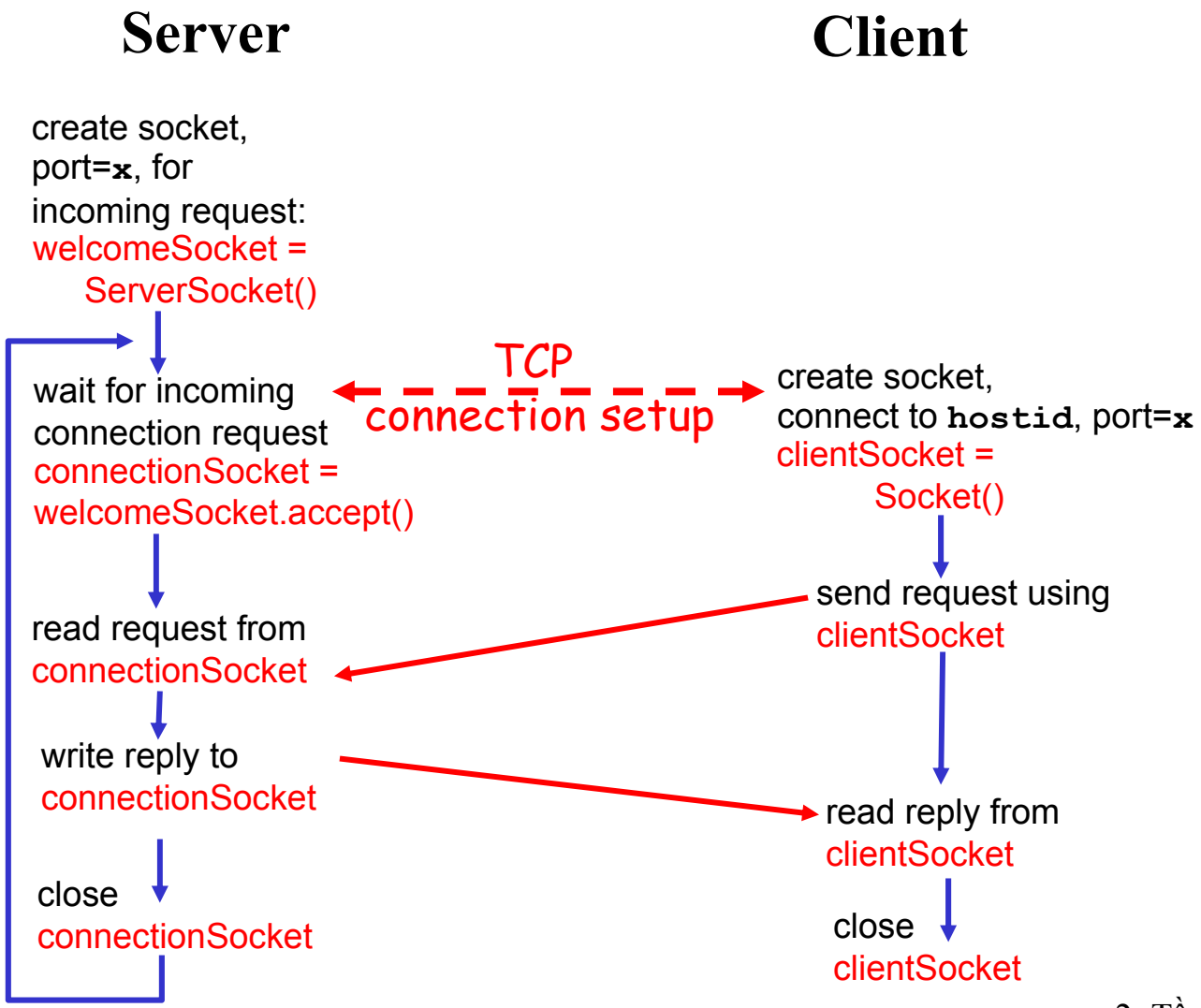
Lập trình Socket TCP

Ví dụ Ứng dụng client-server:

- Client đọc một dòng kí tự từ input chuẩn (**inFromUser** stream), gửi tới server qua socket (**outToServer**)
- server đọc dòng kí tự từ socket
- server biến đổi dòng ký tự đó (chữ thường thành chữ hoa) và gửi trả về cho client.
- client đọc dòng ký tự đã biến đổi từ socket, in ra (**inFromServer**)



Tương tác Socket Client/server : TCP



ServerSocket

- **ServerSocket**()
  - Tạo ra một socket lắng nghe kết nối từ client
- **ServerSocket(int port)**
  - Tạo ra một socket lắng nghe kết nối từ client tại cổng Port
- **ServerSocket(int port, int backlog)**
- **ServerSocket(int port, int backlog, InetAddress bindAddr)**

- **bind(SocketAddress endpoint)**
- **bind(SocketAddress endpoint, int backlog)**
- **Socket accept()**
- **close()**

Socket

- Socket(InetAddress address, int port)
- Socket(InetAddress address, int port, InetAddress localAddr, int localPort)
- Socket(String host, int port)
- bind(SocketAddress bindpoint)
- connect(SocketAddress endpoint)
- connect(SocketAddress endpoint, int timeout)
- InputStream get InputStream()
- OutputStream getOutputStream()
- close()

Ví dụ Java Client (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {

    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;

        BufferedReader inFromUser =
          new BufferedReader(new InputStreamReader(System.in));

        Socket clientSocket = new Socket ("hostname", 6789);

        DataOutputStream outToServer =
          new DataOutputStream(clientSocket.getOutputStream());
```

Tạo input stream

Tạo client socket, kết nối tới server

Tạo output stream, đính kèm vào socket

Ví dụ về Java Client (TCP)

Tạo input stream, đính kèm vào socket

```
BufferedReader inFromServer =
  new BufferedReader(new
  InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();

outToServer.writeBytes(sentence + '\n');

modifiedSentence = inFromServer.readLine();

System.out.println("FROM SERVER: " + modifiedSentence);

clientSocket.close();

    }
}
```

Gửi dòng kí tự đến server

Đọc dòng kí tự (đã biến đổi) do server gửi về

Ví dụ về Java Server (TCP)

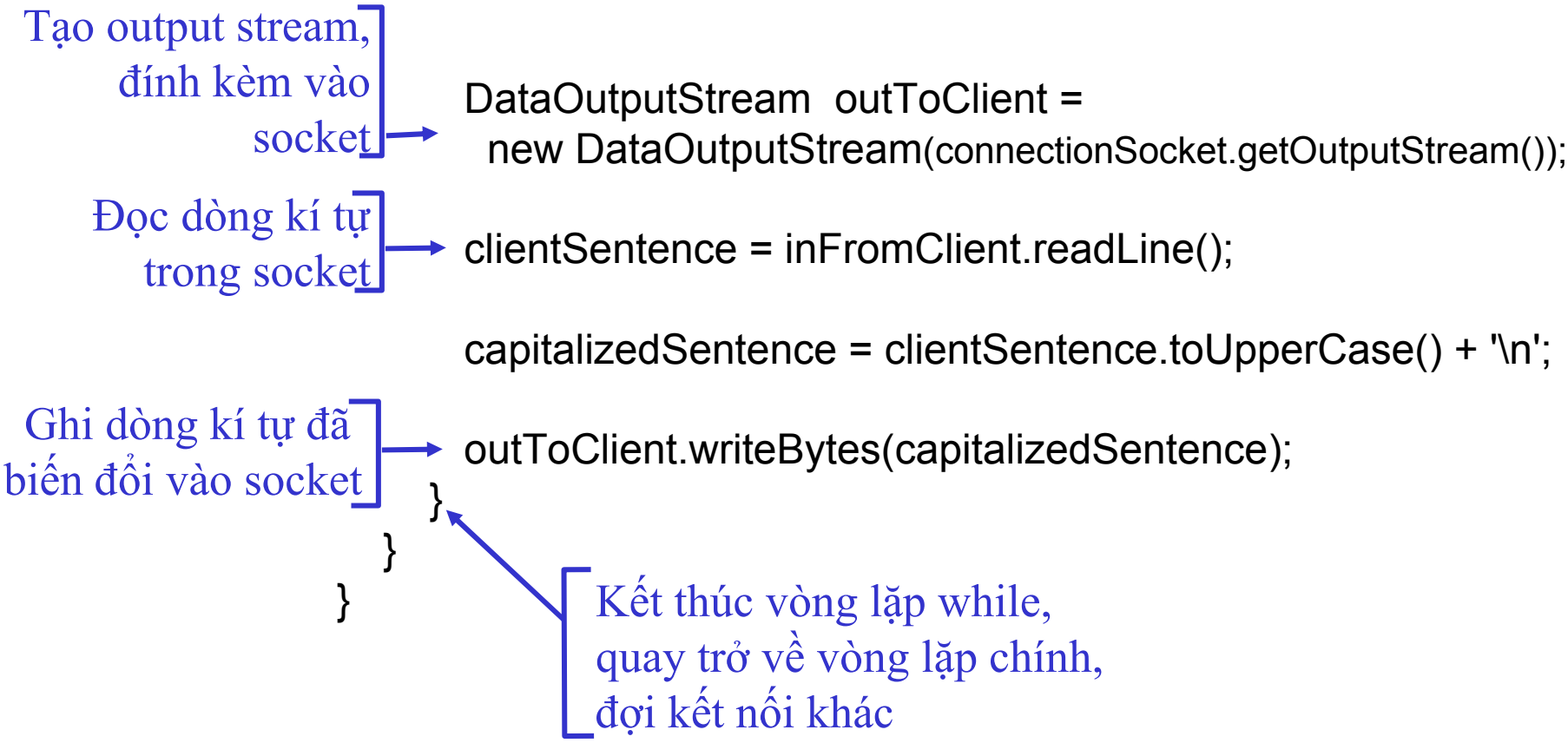
```
import java.io.*;
import java.net.*;

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
     String clientSentence;
     String capitalizedSentence;

     ServerSocket welcomeSocket = new ServerSocket(6789);

     while(true) {

        Socket connectionSocket = welcomeSocket.accept();

        BufferedReader inFromClient =
          new BufferedReader(new
          InputStreamReader(connectionSocket.getInputStream()));
```

Tạo Socket để đợi ở cổng 6789

Đợi đến khi có socket từ client gửi đến

Tạo input stream, đính kèm vào socket

Ví dụ về Java Server (TCP)

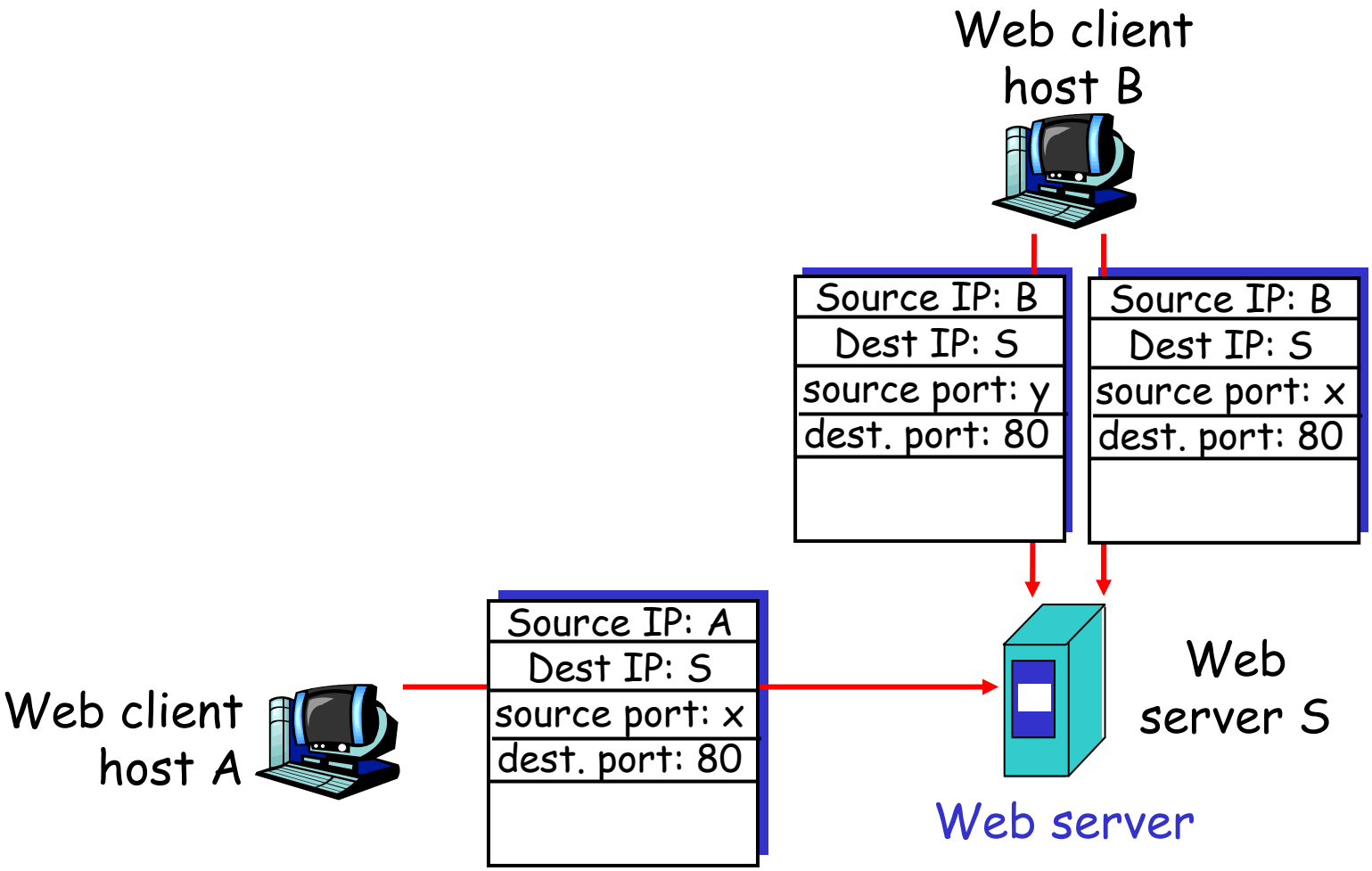


Phân kênh trong Hướng nối

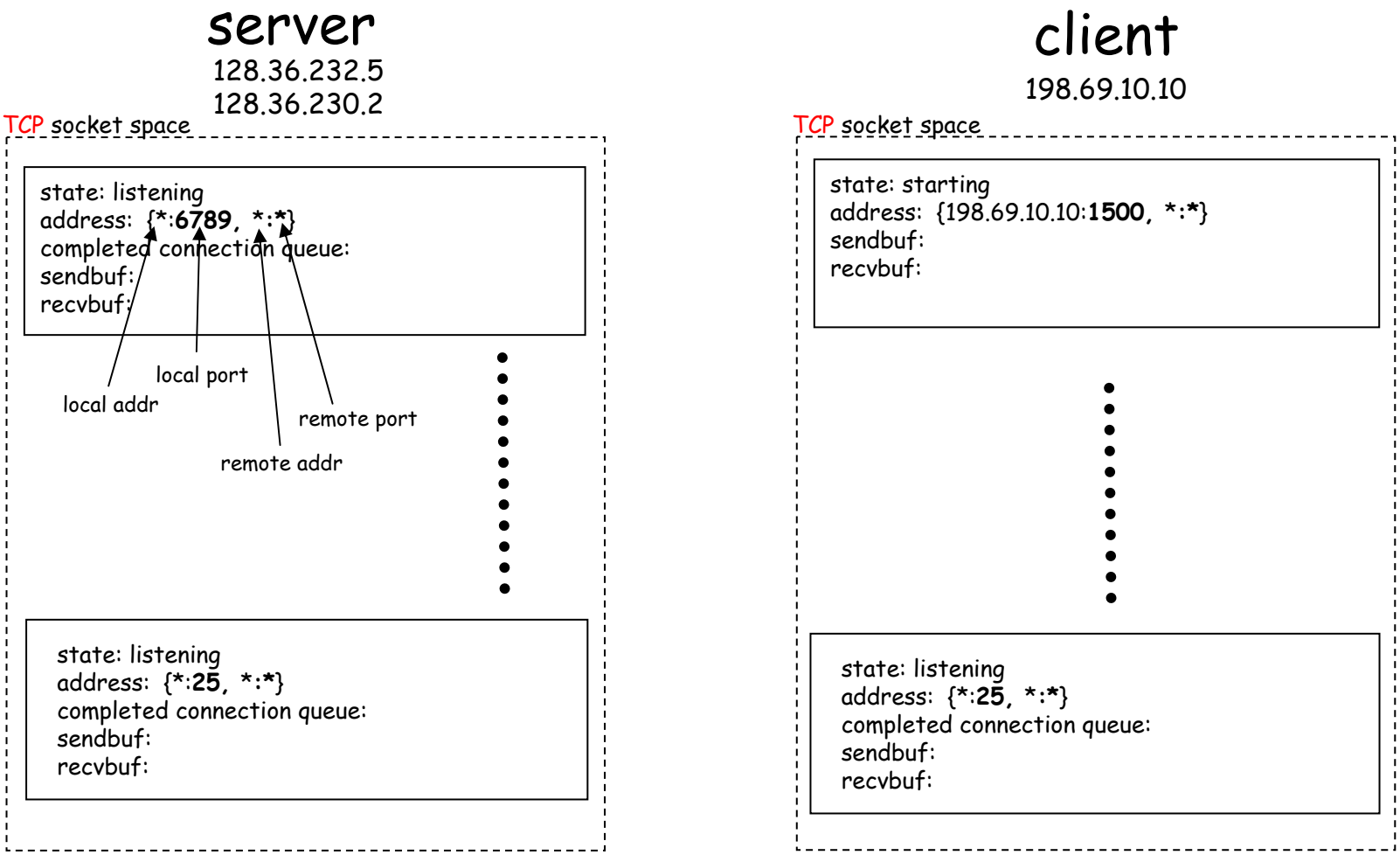
- TCP socket được xác định bởi 4 thành phần:
  - Địa chỉ IP gửi
  - Cổng tiến trình gửi
  - Địa chỉ IP nhận
  - Cổng tiến trình nhận

- Máy tính phía nhận sẽ sử dụng cả 4 thành phần này để chuyển segment đến socket phù hợp
  - Server có thể đồng thời hỗ trợ nhiều socket. Các kết nối khác nhau được tự động chuyển cho các socket tương ứng

Hướng nối : Phân kênh (tiếp)

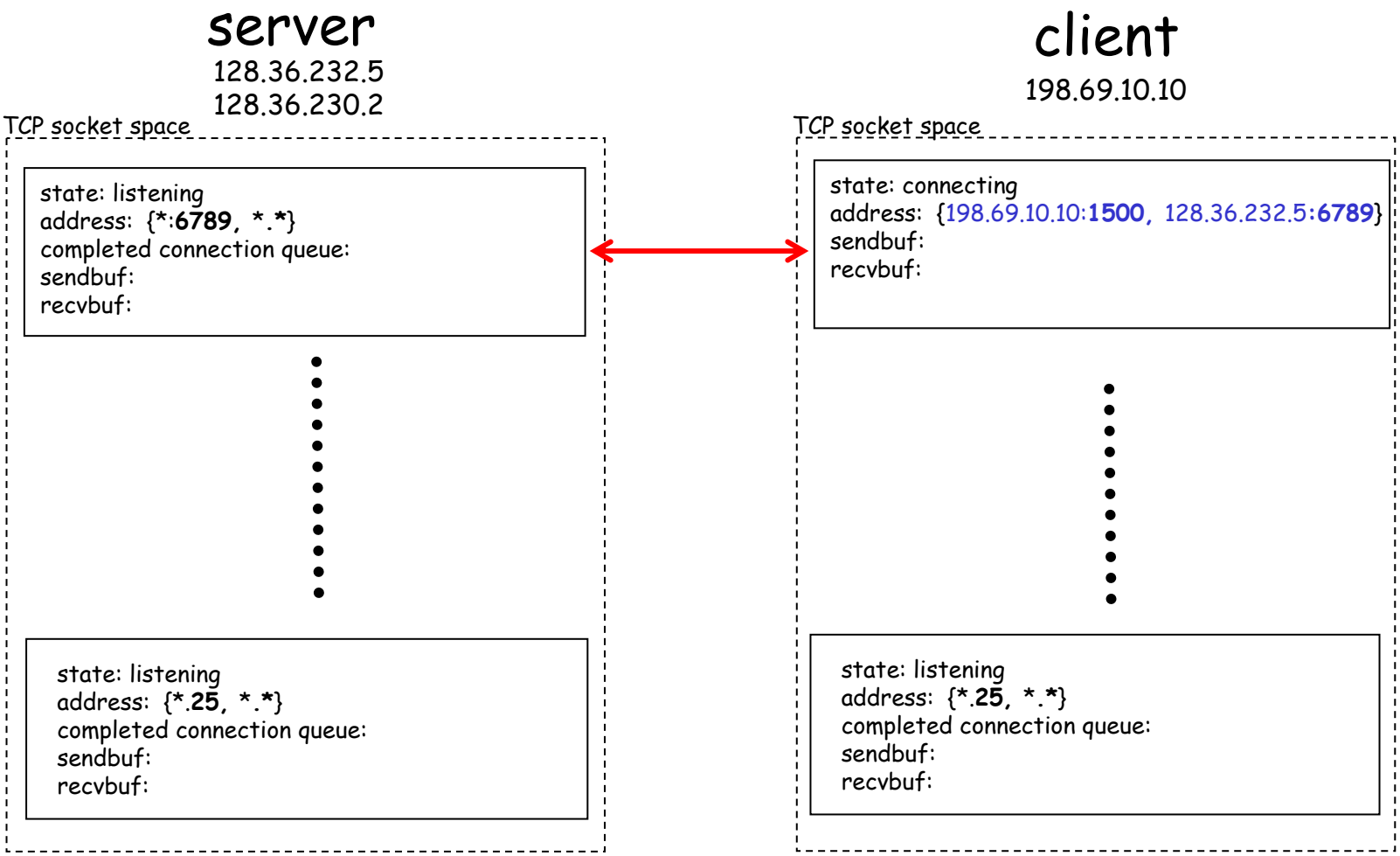


TCP sẽ làm như thế nào

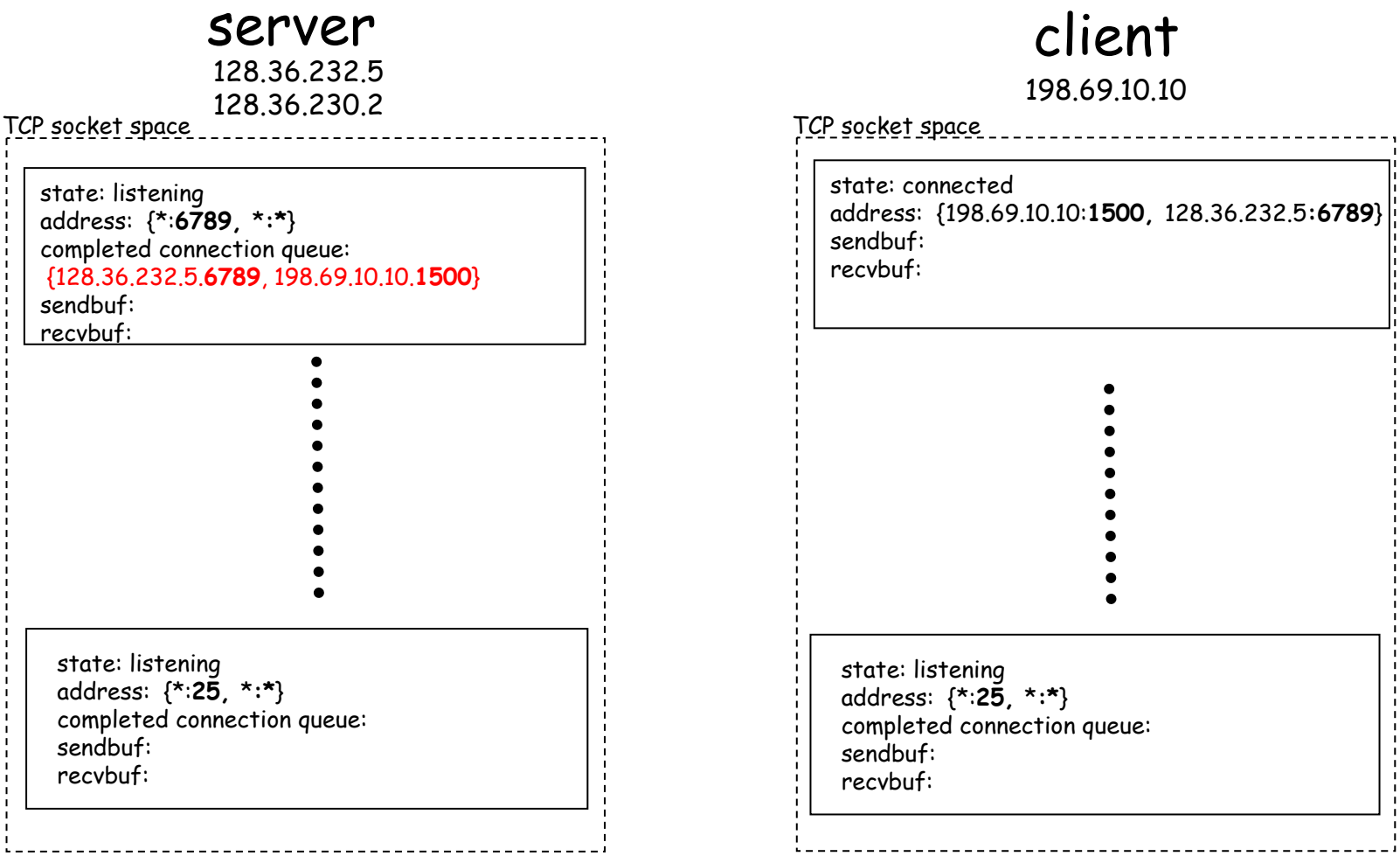


%netstat --tcp -a -l -n

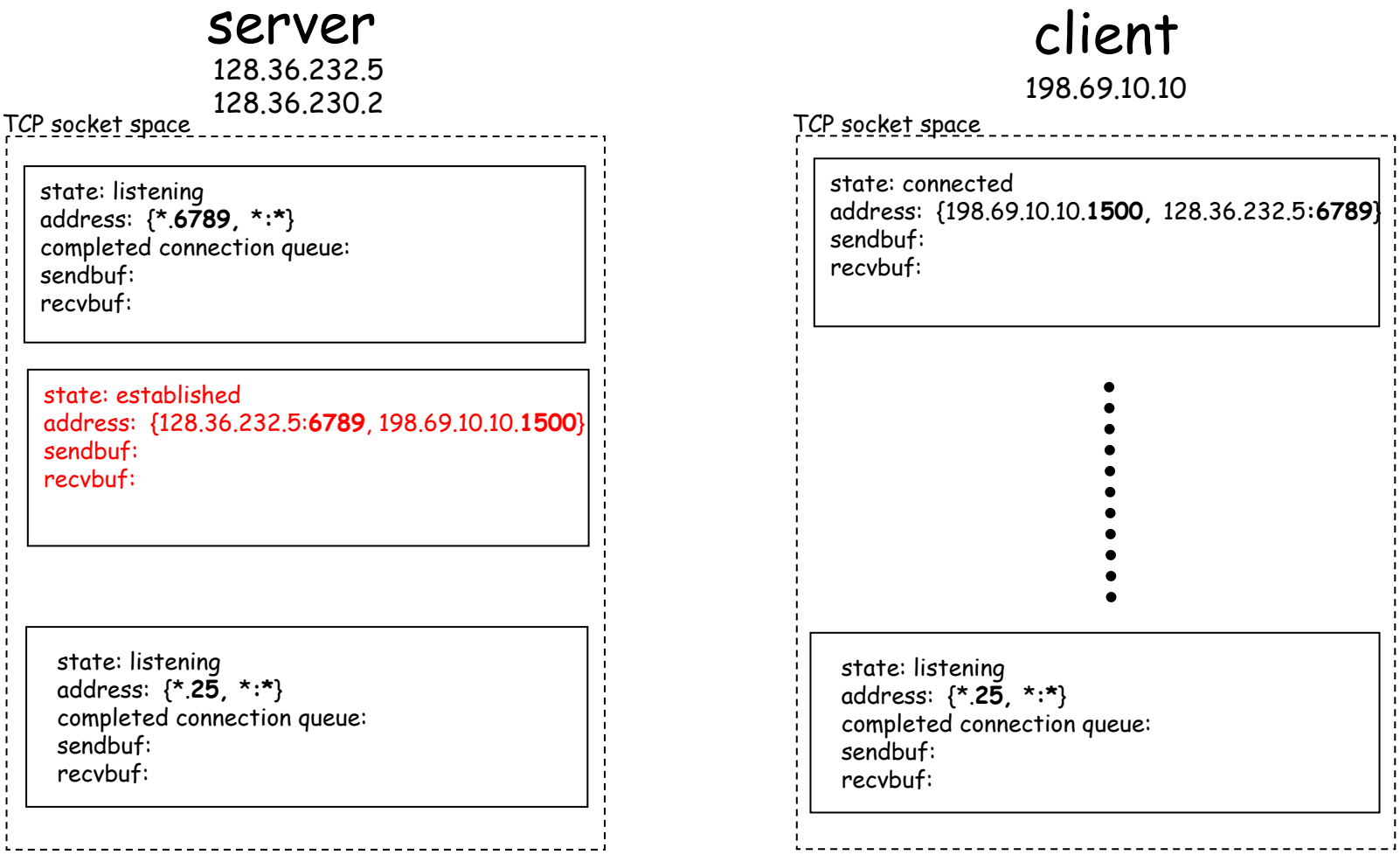
Ví du : Client khởi tạo Kết nối



Kết nối TCP thành công



Ví dụ Server chấp nhận kết nối accept()



Phân kênh cho gói tin dựa trên (IP gửi, port gửi, IP nhận, port nhận)

Packet được gửi cho socket phù hợp nhất!

Lập trình Socket UDP

UDP: không thiết lập kết nối giữa client và server

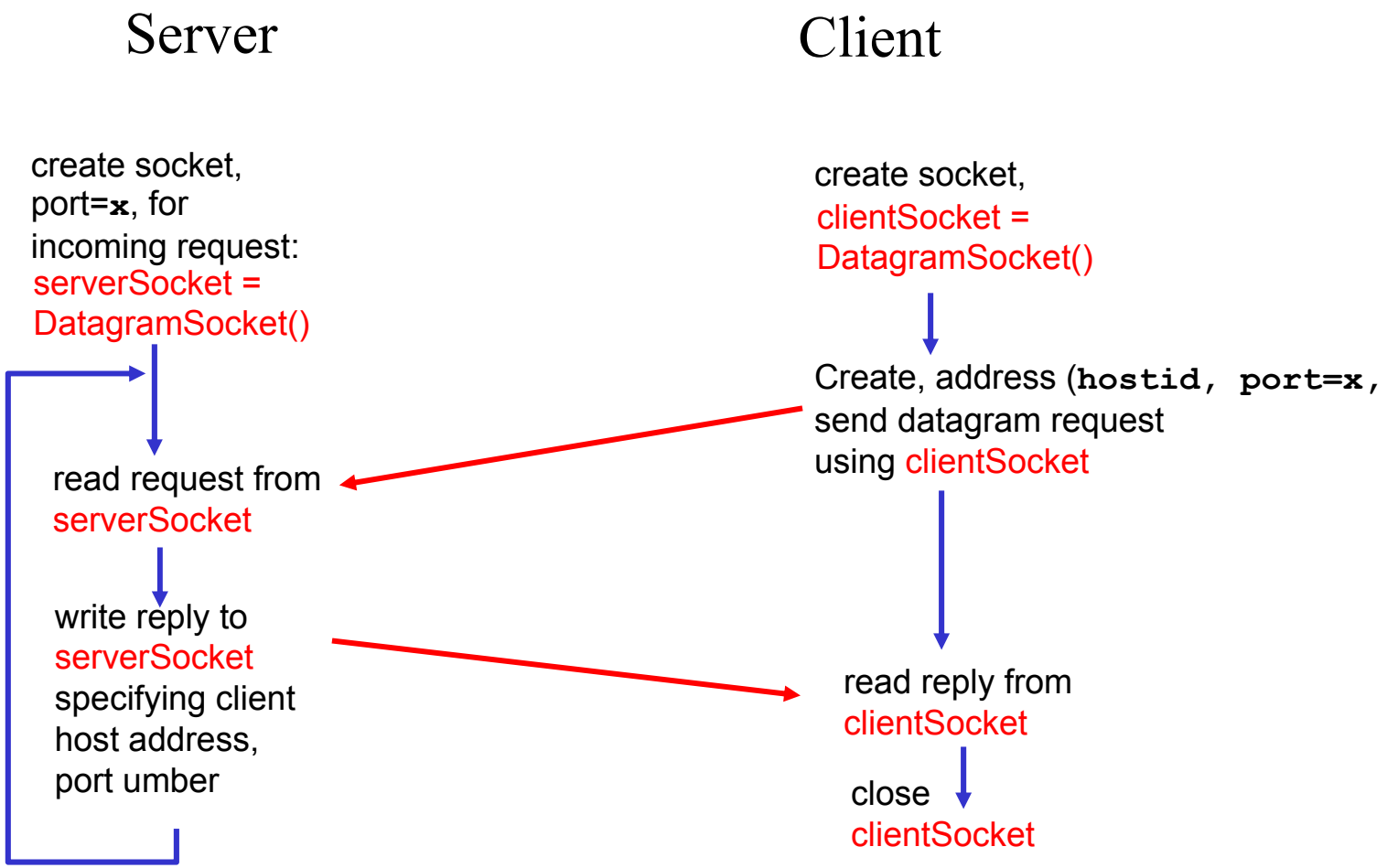
- Không "bắt tay".
- Bên gửi phải xác định chính xác địa chỉ IP và cổng của tiến trình nhận.
- Server xác định địa chỉ IP và cổng của bên gửi từ UDP datagram nhận được.

UDP : dữ liệu truyền đi có thể đến đích không theo đúng thứ tự hay mất mát.

Với Lập trình Ứng dụng

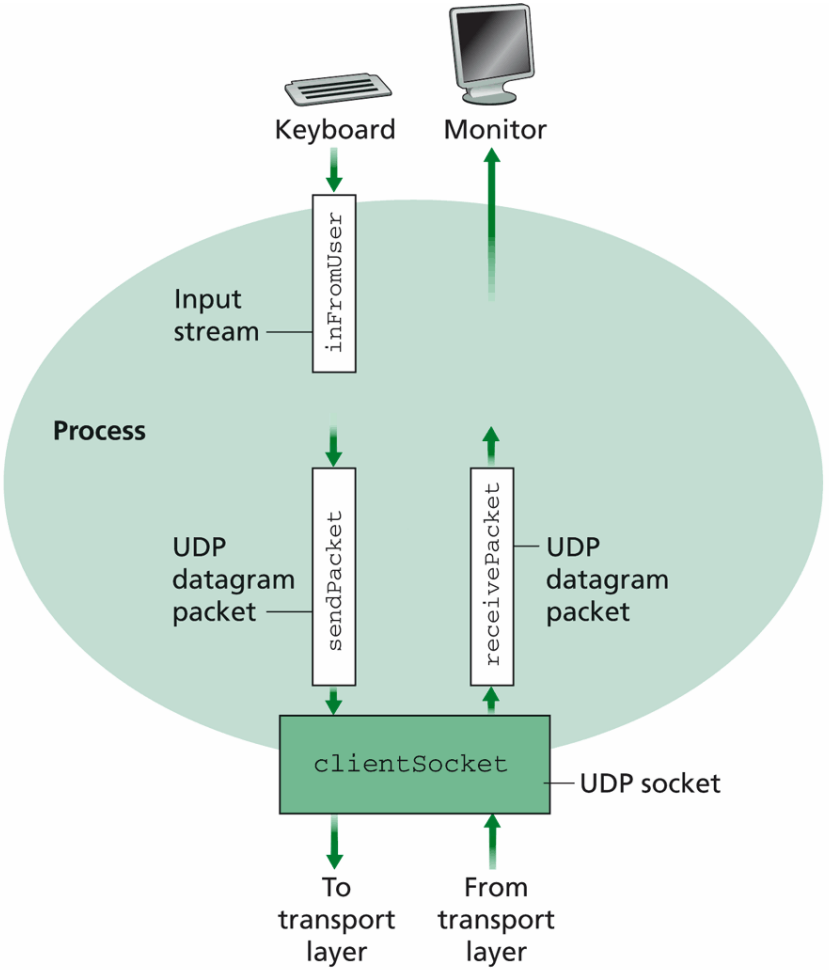
UDP cung cấp dịch vụ truyền dữ liệu theo từng nhóm byte (datagram) không tin cậy

Tương tác Socket Client/Server : UDP



Example: UDPClient.java

- UDP client đơn giản đọc input từ bàn phím, gửi qua server và đợi server trả kết quả về.



Ví dụ Java Client (UDP)

```
import java.io.*;
import java.net.*;

class UDPClient {
    public static void main(String args[]) throws Exception
    {

      BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));

      DatagramSocket clientSocket = new DatagramSocket();

      InetAddress IPAddress = InetAddress.getByName("hostname");

      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];

      String sentence = inFromUser.readLine();

      sendData = sentence.getBytes();
```

Tạo input stream

Tạo client socket

Chuyển hostname sang địa chỉ IP sử dụng DNS

Ví dụ Java Client (UDP)

Tạo datagram cùng với dữ liệu,độ dài, địa chỉ IP, cổng

Gửi datagram tới server

Đọc datagram gửi về từ server

```
                DatagramPacket sendPacket =
                  new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

                clientSocket.send(sendPacket);

                DatagramPacket receivePacket =
                   new DatagramPacket(receiveData, receiveData.length);

                clientSocket.receive(receivePacket);

                String modifiedSentence =
                    new String(receivePacket.getData());

                System.out.println("FROM SERVER:" + modifiedSentence);
                clientSocket.close();
                }
            }
```

Ví dụ Java Server (UDP)

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws Exception
    {
      DatagramSocket serverSocket = new DatagramSocket(9876);

      byte[] receiveData = new byte[1024];
      byte[] sendData  = new byte[1024];

      while(true)
        {
          DatagramPacket receivePacket =
            new DatagramPacket(receiveData, receiveData.length);
          serverSocket.receive(receivePacket);
```

Tạo datagram socket ở cổng 9876

Tạo datagram

Nhận datagram

Ví dụ về Java Server (UDP)

```
String sentence = new String(receivePacket.getData());

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

                String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();

DatagramPacket sendPacket =
   new DatagramPacket(sendData, sendData.length, IPAddress,
                      port);

serverSocket.send(sendPacket);
        }
    }
}
```

Nhận địa chỉ IP và cổng của bên Gửi

Tạo datagram để gửi tới client

Đính datagram vào socket

Kết thúc vòng lặp while, Quay trở về vòng lặp chính, đợi datagram khác đến

Lập trình Socket : Tham khảo

C-language tutorial (audio/slides):

- "Unix Network Programming" (J. Kurose),

HTTP://manic.cs.umass.edu/~amldemo/courseware/intro.

Java-tutorials:

- "All About Sockets" (Sun tutorial), HTTP://www.javaworld.com/javaworld/jw-12-1996/jw-12-sockets.html
- "Socket Programming in Java: a tutorial," HTTP://www.javaworld.com/javaworld/jw-12-1996/jw-12-sockets.html

Phần 2 : TỔNG KẾT

Đã học xong tầng ỨNG DỤNG !

- Các yêu cầu của Dịch vụ tầng Ứng dụng:
  - Sự tin cậy, Băng thông, Độ trễ.
- Mô hình Client - Server.
- Các mô hình dịch vụ của tầng giao vận trên Internet
  - Hướng kết nối, tin cậy: TCP
  - Không tin cậy, datagram: UDP

- **Các Giao thức đặc trưng:**
  - HTTP
  - FTP
  - SMTP, POP3
  - DNS

- **Lập trình Socket**
  - Giao tiếp client/server
  - Sử dụng TCP, UDP socket

Phần 2 : TỔNG KẾT

Trong chương này, chú trọng về GIAO THỨC !

- **Chủ yếu trao đổi thông điệp Request/Reply:**
  - Client yêu cầu thông tin hoặc dịch vụ.
  - Server trả lại dữ liệu cùng mã trạng thái.
- **Định dạng Thông điệp:**
  - Tiêu đề : các trường lưu giữ thông tin về dữ liệu.
  - Dữ liệu: thông tin được trao đổi.

Nhiều Dịch vụ trái ngược :

- Thông điệp mang điều khiển vs. thông điệp mang dữ liệu.
  - in-based, out-of-band
- Tập trung và Phân tán.
- Không trạng thái và Lưu trạng thái.
- Truyền thông điệp Tin cậy và Không tin cậy.
- An ninh : Kiểm chứng.